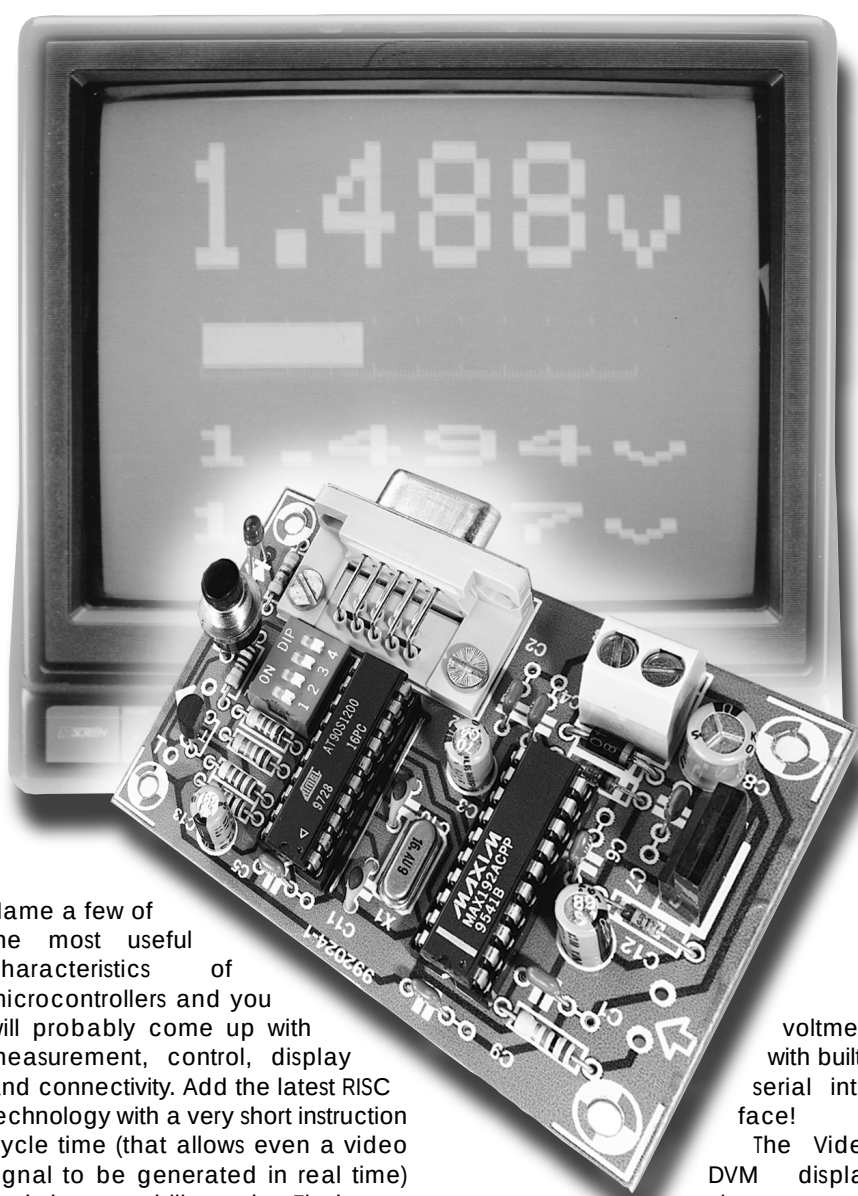


Digital voltmeters are popular tools for anyone wishing to measure electrical voltages. The circuit described here is, we feel, unique because of its combined features. The DVM has two outputs, one serial and one for direct connection of a TV or a monitor.

Design by A. Ricci Bitti

video DVM

measurement values on monitor, TV or PC



Name a few of the most useful characteristics of microcontrollers and you will probably come up with measurement, control, display and connectivity. Add the latest RISC technology with a very short instruction cycle time (that allows even a video signal to be generated in real time) and the capability to be Flash programmed without costly equipment and — *voilà* — you have the Video-DVM, a cheap, funny, giant-size display

voltmeter with built in serial interface!

The Video-DVM displays voltages on your TV screen, both with giant digits and analogue-bar; records maximum and minimum peak; it also

sends measured data to a personal computer through a serial interface. Its block diagram is shown in **Figure 1**. Here are its main features:

- ✓ An analogue input capable to measure voltages in the range 0 to 4.096 V;
- ✓ A video display output capable to direct drive any ordinary TV set via the SCART (VCR) input;
- ✓ A serial data output, to log data on a personal computer;
- ✓ Needs only a single +5V, 15 mA typ. supply.

With its big on-screen display, the Video-DVM is great to show to a large audience practically anything that can be converted to volts, from temperature in a serious physics experiment in the classroom to a *love-meter* for a party.

But the Video-DVM is a lot more than a funny toy. It is really cheap and easy to modify to suit your needs, letting everyone to experiment with video signals and data display. The programming itself meets the severe constraints of real time direct video synthesis, digital conversion, serial data timing, and reduced code size. The analogue to digital converter can be easily connected to a variety of sensors, or can be replaced with new code to display time or count pulses. Last but not least, thanks to its serial datastream the Video-DVM can also be used as a simple data logger.

All the hard work of interrupt-driven TV/monitor picture generation, serial data routines, ADC converter driving, etc. is ready-programmed, enabling anyone to modify the code in small steps to suit any specific need or brilliant idea.

In other words, building the Video-

DVM can be very instructive, a major characteristic in this day and age of hyper-specialized ICs that leave almost nothing to fantasy and exploration.

The hardware circuit

As you can see from the circuit diagram in **Figure 2**, the circuit is built around the new Atmel's AVR 90S1200 microcontroller and Maxim's MAX192 analogue to digital converter (ADC).

Despite to its 16-MHz clock, Flash program memory and internal EEPROM, the AVR is relatively cheap, and therefore ideally suited for hobbyists. What's more, all the software needed (including a powerful simulator) is available for free from Atmel's web site at www.atmel.com.

The chip can be reprogrammed at least 1,000 times in a flash, using one of the many programmers already available (some are advertised on Elektor's pages). The detailed flash programming protocol is available on the web site, too.

The chip has 32 bytes of RAM, 64 bytes of EEPROM, 512 words of program memory and an 8 bit timer. The instruction set is concise and very well balanced, and thanks to an Harvard RISC architecture even a complex task like the one described here is accomplished using no more than about 400 instructions.

The instruction cycle time is very short, enabling video signals to be built by software.

A simple two-bit asymmetrical DAC built around three resistors supplies the composite video signal at a standard level of $1 V_{pp}$ to the input of any TV set with a SCART connector or an AV (VCR) input. The serial datastream (ASCII data at 1200 baud, no parity, 8 data bits, 1 stop bit) is also generated by software. The level is TTL: most personal computers work with TTL levels as well as with standard RS232 levels, provided that the connecting cable is not too long. Because of this I thought a voltage converter like the MAX232 not necessary. If you want, you can add it externally; in that case you must invert the polarity, inverting the 'set bit' and 'clear bit' instructions in the code (the position is clearly marked in the listing).

All input pins have internal pullups and high current sink capacity, so the 'max-min clear' button and the three decimal point selection DIP switches or jumpers may be connected directly between the pins and ground.

An LED, flashing each time a measurement is completed, is connected to an output pin.

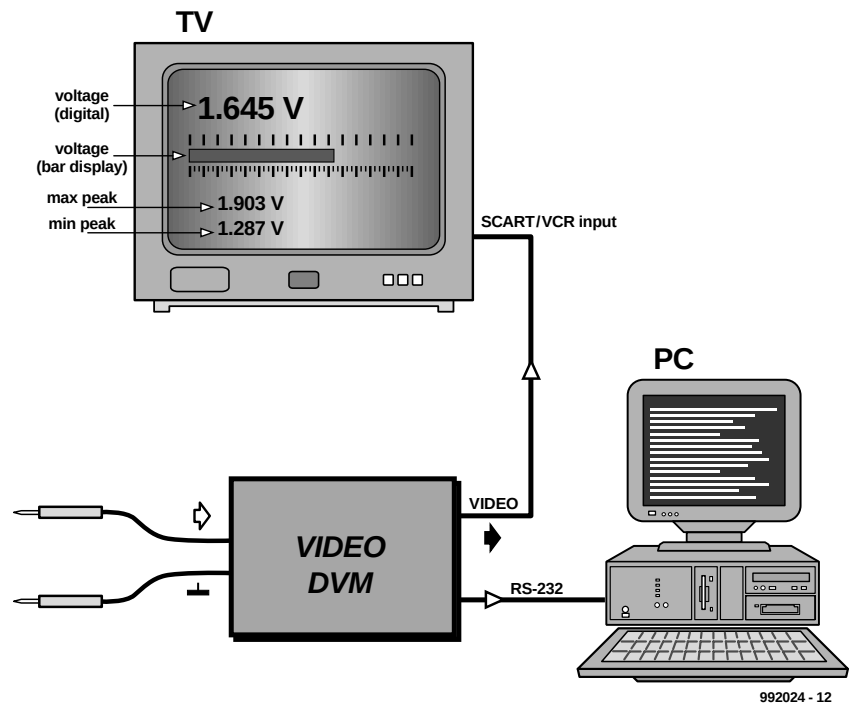


Figure 1. Block diagram of the digital voltmeter with serial and video output.

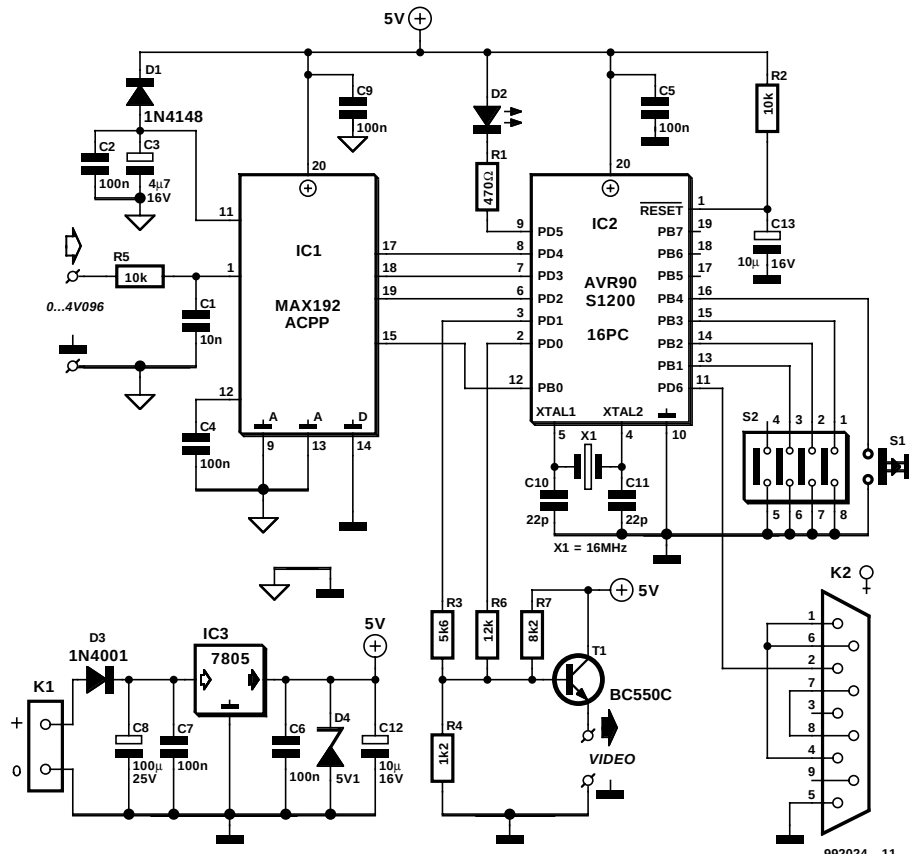
The ADC concept

The voltage is read using a 10-bit serial ADC converter type MAX192ACPP with its own voltage reference set at 4.096 V. Technical details on this interesting chip

may be found on the Datasheets in the April 1999 issue of *Elektor Electronics*.

Four I/O lines are needed for data transfers (Data input, Data output, Chip enable, Serial clock). Even if it is specified for 10 bits, the ADC supplies two

Figure 2. From theory to practice: a microcontroller and an A-D converter.



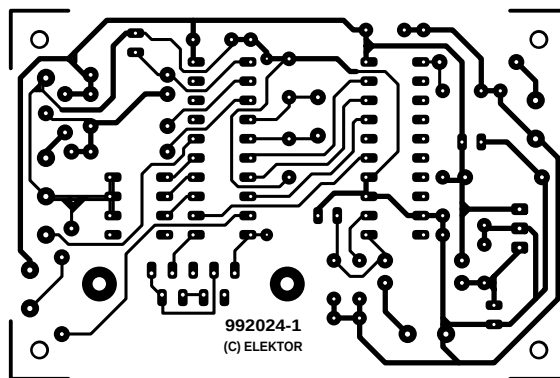
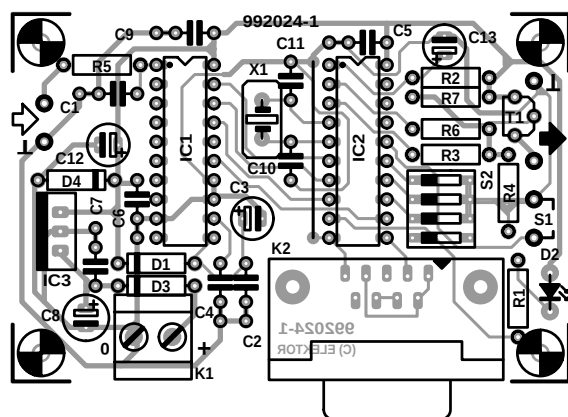


Figure 3. Printed circuit board design (board available ready-made).

COMPONENTS LIST

Resistors:

R1 = 470Ω
R2, R5 = 10kΩ
R3 = 5kΩ
R4 = 1kΩ
R6 = 12kΩ
R7 = 8kΩ

Capacitors:

C1 = 10nF, ceramic
C2, C4-C7, C9 = 100nF, ceramic
C3 = 4μF 16V radial
C8 = 100μF 25V radial

C10, C11 = 22pF
C12, C13 = 10μF 16V radial

Semiconductors:

D1 = 1N4148
D2 = LED, high efficiency
D3 = 1N4001
D4 = 5V1 400mW zener diode
T1 = BC550C
IC1 = MAX192ACPP
IC2 = AVR90S1200-16PC (order code)
IC3 = 7805

Miscellaneous:

S1 = pushbutton, 1 make contact

S2 = 3-way or 4-way DIP switch
X1 = 16MHz quartz crystal
K1 = 2-way PCB terminal block, raster 5mm
K2 = 9-way sub-D socket (female), angled pins, PCB mount
PCB, order code 992024-1

The CD-ROM 'μP/μC hard & software 97/98' (ISBN 90 5381-086-2) available from the Publishers contains the complete source code listing for this project. The file location is /Gb/03.

more 'sub LSB' bits: this means that 12 bits are effectively read from the ADC, although only 10 bits have the guaranteed precision. Nonetheless, I found these two extra bits very precise. All 12 bits are shown on the display, covering the range from 0.000 to 4.095 volts with direct reading in millivolts. Only one of the eight inputs available on the MAX192 is used, but the software driver is prepared to read any input you want. With minor changes, you can set up the Video-DVM to measure four voltages; three with digital display and one as an analogue bar.

Direct video synthesis.

From the hardware point of view, the hardware involved in video synthesis is exceptionally simple: two output pins and three resistors, allowing for four signal levels to be generated (sync, black, white, light grey). From the software point of view, video synthesis requires a very fast instruction cycle (here, 62.5 ns) and a carefully timed, instruction-balanced, tweaked and manually optimised code.

In order to achieve proper generation of the complex video signal, a robust timing system is absolutely necessary. Even a single 16-MHz clock

cycle (62.5 ns) delay is clearly visible on the display, so you have to think twice before putting down even a single instruction.

First of all, we have to choose a suitable timebase: choosing the raster line duration (64 μs) as time base let us to build the entire frame a line after another (the non-interlaced frame consists of 312 lines), as well as placing easily the horizontal sync pulse (a few microseconds at start of each video line) and the vertical pulse (a few lines at start of each frame).

The only timer available on the micro is capable of generating repetitive interrupts every 16 μs without the need of reloading; that is, four interrupts for every raster line.

At each fourth interrupt we make a new video line: to do this, each time the interrupt routine is executed, a counter is incremented and every four interrupts we are (almost) at the start of a new line. "Almost", because we have to account for the time needed to service the interrupt routine upon interrupt request. This time is not constant, depending on the instruction being executed at the time of the interrupt generation. Some instructions have a longer execution time than others, so the interrupt service time can vary in an unpredictable way, distorting

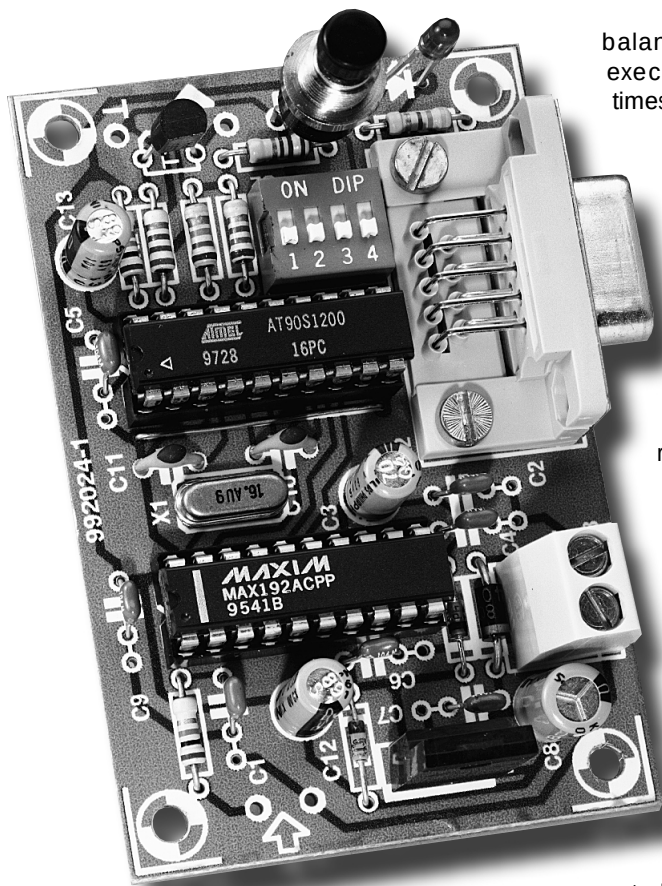
the display.

The best workaround I found for the problem is to put the micro in sleep mode just before the fourth, critical interrupt occurs: the next interrupt will then wake up the micro with constant, known timing.

The two output pins are labelled CsyncBit (composite sync) and VideoBit. Pulling both Csync and VideoBit to logic-low, the video signal is at 0 volts (sync level). With only CsyncBit high, we get the black level; with both CsyncBit and VideoBit high, we get a white display.

Not surprisingly, all the video generation is based on the timer interrupt routine. Every fourth interrupt a new line is started with the sync pulse; then the repetitive timekeeping (counting of lines, serial communications) routines are executed. All this work is done so quickly that we have to add a delay loop to wait for the start of the visible portion of the video line.

At this point, a multiple jump structure is executed to determine what kind of line we are at. The jumps are based on line number (vertical position), so that the micro can decide if the current line will show the current voltage (in which case the character display routine is called), or the analogue bar (bar



balance execution times! When executing branches, execution times associated with every possible path must be carefully considered. Fortunately for you, all the code needed to handle the video signal is ready made and works by itself under interrupt control.

Feel free to modify or add code to the main program to suit your requirements. Even the multiple jump structure that selects the display contents is not too difficult to understand, so you can easily build hundreds of different displays, with characters of any size or with multiple analogue bars.

All routines and variables have long names which should be self-explanatory.

The code is heavily commented. Please remember that only one subroutine call is allowed at a time.

Custom fonts

You can redefine fonts to any shape you like; this is particularly useful for the measurement unit. Instead of the volt symbol (V) you can display any other letter or shape that fits in a 8x5 pixel matrix. Font shapes are stored in EEPROM according to the definition file FONTS.INC. The file is automatically included during assembly and results are compiled to VIDEOVM.EEP; *do not forget to separately program the EEPROM once you have programmed the FLASH part.*

Fonts are stored rotated 90 degrees clockwise, into 5 successive bytes.

Building the circuit

All the components used are cheap and easy to find. The MAX192 has equivalents with the same data protocol with different resolutions.

If your ordinary distributor does not have the Atmel AT90S1200-16PC yet, because this component is relatively new, it can be requested at a number of Elektor's advertisers who also sell programmers for the Atmel flash micros.

The copper track layout and component mounting plan are shown in **Figure 3**. The link to the computer is made via a standard 9-way sub-D connector.

Very likely you will have your preferred sensorware connected to the inputs: be sure that the range never exceeds 4.096 V or goes below 0 V, placing a series resistor (about 1k) and crowbar diodes if that range can be exceeded.

Once powered up, the flashing LED will tell you that the microcontroller is working just fine. The LED flashes once per conversion.

Now is the time to connect the Video-DVM to the TV set: for TVs equipped with a SCART connector the pins for video input are 20 (video) and 17 (ground). The TV must be disconnected from the mains while you insert the SCART plug.

Switch the TV to AV input mode using the remote control: older TV sets may not have this capability, in that case the AV input selection can be forced by pulling up the SCART pin 8: simply install a 1-k Ω series resistor from the pin to +5-12V.

You can now fit the jumpers for the selection of the decimal point position (**Table 1**), play around with various voltages to the ADC, verify the effect of the max-min clear button, or simply admire your new electronic toy for a while.

The last step is to connect the personal computer. Start the Terminal program in the Accessories folder (Windows 3.11) or the HyperTerminal utility (Windows 95). Set the connection to 1200 baud, no parity, 8 data bits, one stop bit (1200,n,8,1). The standard measurement range is from 0 up to 4.095 V. If a larger range is required, an attenuator should be added. One solution that comes to mind is to fit a resistor in parallel with C1. When higher input voltages are used, however, there is a real risk of the input of the MAX192 being damaged by wrong connections. In that case, the preventive measure is to fit a 5.1-volt zener diode across C1 and the resistor. A measurement range of 0 to 10 V is created, for example, by fitting a 6.9-k Ω resistance (5.6 k Ω fixed resistor plus 2.2 k Ω preset) across C1.

(992024-1)

Table 1.
DIP switch settings (S2)

1	2	3	display
OFF	OFF	OFF	xxx.x
OFF	OFF	ON	xx.xx
OFF	ON	OFF	x.xxx
OFF	ON	ON	.xxxx
ON	OFF	OFF	xx.xx
ON	OFF	ON	x.xxx
ON	ON	OFF	.xxxx
ON	ON	ON	xxxx